

The Portable Executable Journey

Version 1.0

May-2024

By Dr. Shlomi Boutnaru

```
00000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 40 00 00 00 00 00 00 MZ.....@.....
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 18 01 00 00 .....!..L!This program cannot
00000040 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 69 73 20 70 72 6F 67 72 61 60 20 63 61 6E 6E 6F .....t be run in DOS mode...$.
00000060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 .....
00000080 9D 05 89 E0 D9 B4 E7 B3 D9 B4 E7 B3 D9 B4 E7 B3 CD DF E1 B2 D0 B4 E7 B3 CD DF E6 B2 D5 B4 E7 B3 .....
000000A0 D9 B4 E6 B3 EF B4 E7 B3 82 DC E3 B2 D8 B4 E7 B3 82 DC E4 B2 D7 B4 E7 B3 D9 B4 E7 B3 58 B4 E7 B3 .....X.
000000C0 D0 CC 74 B3 CE B4 E7 B3 CD DF EA B2 2D B1 E7 B3 CD DF E2 B2 C2 B4 E7 B3 CD DF E3 B2 A3 B4 E7 B3 .....t.
000000E0 CD DF E4 B2 E4 B4 E7 B3 CD DF E7 B2 D8 B4 E7 B3 CD DF 18 B3 D8 B4 E7 B3 CD DF E5 B2 D8 B4 E7 B3 .....
00000100 52 69 63 68 D9 B4 E7 B3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 50 45 00 00 64 86 21 00 Rich.....PE..d.l.
00000120 29 DA 31 AC 00 00 00 00 00 00 00 F0 00 22 00 0B 02 0E 14 00 EE 8B 00 00 7E 1B 00 00 A0 48 00 .....).1.....H.
00000140 10 20 99 00 00 10 00 00 00 00 00 40 01 00 00 00 00 10 00 00 00 02 00 00 0A 00 00 00 0A 00 00 .....@.....E.....A.
00000160 0A 00 00 00 00 00 00 00 60 04 01 00 08 00 00 45 0E A6 00 01 00 60 41 00 00 08 00 00 00 00 00 .....`.....
00000180 00 20 00 00 00 00 00 00 10 00 00 00 00 00 10 00 00 00 00 00 10 00 00 00 00 00 10 00 00 00 .....@.....h.....z.
000001A0 00 40 13 00 7E 8D 01 00 30 16 13 00 68 01 00 00 00 00 01 FC B0 03 00 00 90 0C 00 1C 7A 06 00 .....p%.
000001C0 00 BE A5 00 70 25 00 00 00 C0 03 01 CC 50 00 00 70 0B 01 00 54 00 00 00 00 00 00 00 00 00 00 .....0[.....
000001E0 00 00 00 00 00 00 00 00 00 00 00 00 00 30 5B 00 00 18 01 00 00 00 00 00 00 00 00 00 00 .....@.....H.idata...
00000200 00 10 13 00 20 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....".
00000220 2E 72 64 61 74 61 00 00 60 77 0C 00 00 10 00 00 00 78 0C 00 00 08 00 00 00 00 00 00 00 00 .....~...@.....@...@
00000240 00 00 00 00 40 00 00 48 2E 70 64 61 74 61 00 00 1C 7A 06 00 00 90 0C 00 00 7C 06 00 00 80 0C 00 PROTDATA.....
00000260 00 00 00 00 00 00 00 00 40 00 00 48 2E 69 64 61 74 61 00 00 C2 20 00 00 10 13 00 .....@.....H.idata...
00000280 00 22 00 00 00 FC 12 00 00 00 00 00 00 00 00 00 00 00 40 00 00 48 2E 65 64 61 74 61 00 00 .....".
000002A0 7E 8D 01 00 00 40 13 00 00 8E 01 00 00 1E 13 00 00 00 00 00 00 00 00 00 00 40 00 00 40 .....~...@.....@...@
000002C0 50 52 4F 54 44 41 54 41 01 00 00 00 D0 14 00 00 02 00 00 AC 14 00 00 00 00 00 00 00 00 00 PROTDATA.....
000002E0 00 00 00 00 40 00 00 48 47 46 49 44 53 00 00 3C 8C 00 00 E0 14 00 00 8E 00 00 AE 14 00 .....@.....HGFIIDS...<...
00000300 00 00 00 00 00 00 00 00 40 00 00 42 50 61 64 31 00 00 00 00 90 0A 00 00 70 15 00 .....@.....BPad1.....p.
00000320 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 42 2E 74 65 78 74 00 00 00 .....B.text...
00000340 E9 CE 3C 00 00 20 00 00 D0 3C 00 00 3C 15 00 00 00 00 00 00 00 00 00 20 00 00 68 .....<...<...<...h
00000360 50 41 47 45 00 00 00 C6 4E 3C 00 00 D0 5C 00 00 50 3C 00 00 0C 52 00 00 00 00 00 00 00 PAGE...N<...P<...R.....
00000380 00 00 00 00 20 00 00 60 50 41 47 45 4C 4B 00 00 94 4E 02 00 00 20 99 00 00 50 02 00 00 5C 8E 00 .....`PAGELK...N...P...
000003A0 00 00 00 00 00 00 00 00 20 00 00 60 50 4F 4F 4C 43 4F 44 45 8B 04 00 00 70 9B 00 .....`POOLCODE...p.
000003C0 00 06 00 00 AC 90 00 00 00 00 00 00 00 00 00 00 00 20 00 00 68 50 41 47 45 4B 44 00 00 .....hPAGEKD...
000003E0 92 5B 00 00 00 80 9B 00 00 5C 00 00 B2 90 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60 .....[.....
00000400 50 41 47 45 56 52 46 59 FC 20 03 00 00 E0 9B 00 00 22 03 00 00 0E 91 00 00 00 00 00 00 PAGEVRFY.....".....
00000420 00 00 00 00 20 00 00 60 50 41 47 45 48 44 4C 53 56 25 00 00 10 0F 00 00 76 00 00 30 04 00 .....PAGEFIDS...&...&
```

Screenshot taken from TotalPE

Table of Contents

Table of Contents.....	2
Introduction.....	3
DOS Header (struct _IMAGE_DOS_HEADER).....	4
DOS Stub.....	5
NT Headers (struct _IMAGE_NT_HEADERS32/64).....	6
File Header (struct _IMAGE_FILE_HEADER).....	7
Optional Header (struct _IMAGE_OPTIONAL_HEADER32/64).....	8

Introduction

Understanding the PE (Portable Executable) file format is crucial for different tasks such as analyzing malware and reverse engineering. Also, this file format is used by Windows applications, UEFI firmware and even for Linux applications (such as does based on .NET core).

Thus, I wanted to create something that will improve the overall knowledge of the PE file format with writeups that can be read in 1-3 mins. I hope you are going to enjoy the ride.

Lastly, you can follow me on twitter - @boutnaru (<https://twitter.com/boutnaru>). Also, you can read my other writeups on medium - <https://medium.com/@boutnaru>. Lastly, You can find my free eBooks at <https://TheLearningJourneyEbooks.com>.

Lets GO!!!!!!

DOS Header (struct _IMAGE_DOS_HEADER)

Every PE (Portable Executable) binary starts with a MS-DOS (Microsoft Disk Operating System) header. By the way, the PE format is used by: Windows 95 and higher, Windows NT 3.1 and higher, ReactOS and UEFI. It is also used by .NET assemblies¹.

Overall, the first two bytes (aka “Magic”) is “0x5A4D” which is “MZ” in ASCII. The letter “MZ” stands for “Mark Zbikowski”, who is one of the designers on the MS-DOS executable - as shown in the screenshot below, taken using “<https://hexed.it/>” while examining “mspaint.exe”².

Moreover, the “DOS Header” is defined using “struct _IMAGE_DOS_HEADER”³. As we can see the last field “e_lfanew” contains the file address pointing to the beginning of the PE header (in little endian” - as shown in the screenshot below.

Lastly, the Windows loader cares about “e_magic” and “e_lfanew” from “struct _IMAGE_DOS_HEADER”. The other members (like initial instruction pointer, initial stack pointer, number of pages in the file and more) are relevant for MS-DOS when executing the stud program, which follows the DOS header and is going to be detailed in a future writeup⁴.

```
mspaint.exe x
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZË.....
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  7.....@.....
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000030  00 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00  .....°...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..||.}.=!7.L=!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.
00000080  86 7B 9D 70 C2 1A F3 23 C2 1A F3 23 C2 1A F3 23  â{¥pT.<#T.<#T.<#
00000090  CB 62 60 23 F4 1A F3 23 D6 71 F0 22 C1 1A F3 23  T'b`#|.<#Tq="T.<#
000000A0  D6 71 F7 22 D9 1A F3 23 D6 71 F5 22 F2 1A F3 23  Tq≈"j|.<#Tq÷">.<#
000000B0  D6 71 F2 22 DF 1A F3 23 C2 1A F2 23 CA 1F F3 23  Tq≥"■.<#T.≥#L.<#
000000C0  D6 71 FB 22 42 1A F3 23 D6 71 0C 23 C3 1A F3 23  Tqv"B.<#Tq.#|.<#
000000D0  D6 71 F1 22 C3 1A F3 23 52 69 63 68 C2 1A F3 23  Tq±"|.<#RichT.<#
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000000F0  00 00 00 00 00 00 00 00 50 45 00 00 64 86 07 00  .....PE..dâ..
```

¹ <https://wiki.osdev.org/PE>
² <https://medium.com/@boutnaru/the-windows-process-journey-mspaint-exe-paint-3317a8fb3a57>
³ https://github.com/reactos/reactos/blob/master/drivers/filesystems/udfs/Include/ntddk_ex.h#L99
⁴ <https://osandamalith.com/2020/07/19/exploring-the-ms-dos-stub/>

DOS Stub

Just after the “DOS Header”⁵ and before the “NT Headers” we have the “DOS Stub”. The DOS stub is a program which is invoked in case the file is executed in MS-DOS. By default it displays the following message: “This program cannot be run in MS-DOS mode.” - as shown in the screenshot below, taken using “<https://hexed.it/>” while examining “cmd.exe”⁶. Any valid MS-DOS application can act as the DOS stub program, due to that we can also change it if we want⁷.

Moreover, the DOS stub is a 16-bit program (real-mode). Also, in case the stub program is executed the instruction pointer (EIP), stack pointer (ESP), the code segment (CS) and stack segment are initialized based on data stored as part of the DOS header: “e_ip”, “e_sp”, “e_cs” and “e_ss” respectively⁸.

Lastly, we can see in the screenshot below that the opcode “int 0x21” is in use (CD 21). Most of the general functions and services offered by DOS are implemented through this interrupt⁹. In the case of the DOS stub it is used for printing and quitting the program (with an exit code).

```
cmd.exe x
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZÉ.....
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  7.....@.....
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000030  00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00  .....=...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..||..|.=!7.L=!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.
00000080  AB 4F 48 95 EF 2E 26 C6 EF 2E 26 C6 EF 2E 26 C6  %0Hòñ.&=ñ.&=ñ.&=
00000090  E6 56 B5 C6 A9 2E 26 C6 FB 45 25 C7 EC 2E 26 C6  μV| |-.&=√E%|∞.&=
000000A0  FB 45 22 C7 F9 2E 26 C6 EF 2E 27 C6 C6 2F 26 C6  √E" |..&=√E# |μ.&=
000000B0  FB 45 27 C7 EA 2E 26 C6 FB 45 23 C7 E6 2E 26 C6  √E' |Ω.&=√E# |μ.&=
000000C0  FB 45 2B C7 C6 2E 26 C6 FB 45 D9 C6 EE 2E 26 C6  √E+ | |.&=√E| |ε.&=
000000D0  FB 45 24 C7 EE 2E 26 C6 52 69 63 68 EF 2E 26 C6  √E$ |ε.&=Richñ.&=
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000000F0  50 45 00 00 64 86 07 00 0D 19 EE D7 00 00 00 00  PE..dâ....ε|....
00000100  00 00 00 00 F0 00 22 00 0B 02 0E 14 00 10 02 00  = "
```

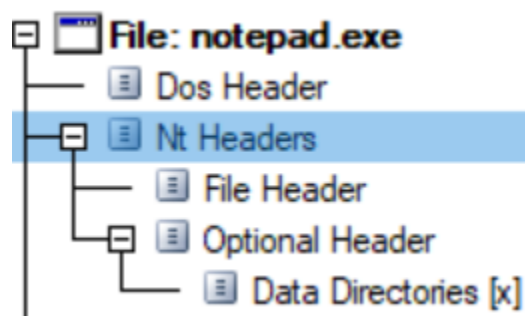
⁵ <https://medium.com/@boutnaru/the-portable-executable-journey-dos-header-ea5b29f15612>
⁶ <https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81>
⁷ <https://learn.microsoft.com/en-us/cpp/build/reference/stub-ms-dos-stub-file-name?view=msvc-170>
⁸ <https://osandamalith.com/2020/07/19/exploring-the-ms-dos-stub/>
⁹ http://bbc.nvg.org/doc/Master%20512%20Technical%20Guide/m512techb_int21.htm

NT Headers (struct _IMAGE_NT_HEADERS32/64)

Pointed from the “DOS Header”¹⁰ and should be after the “DOS stub”¹¹ we have the “NT Headers”. The “NT Header” (aka “PE Header”) is defined in one of two data structures: “struct _IMAGE_NT_HEADERS32”¹² for 32-bit binaries and “struct _IMAGE_NT_HEADERS64”¹³ for 64-bit binaries.

Overall, the data structure is composed of 3 fields: “Signature”, “File Header” (struct _IMAGE_FILE_HEADER) and “Optional Header” (struct _IMAGE_OPTIONAL_HEADER32/64). Signature is “PE\0\0”, which are the letters "P" and "E" followed by two null bytes. File header is a standard COFF header - which we are going to detail in a future writeup. The optional header is a must in the case of an image file and optional only in case of an object file¹⁴.

Lastly, because there are two versions for the “Optional Header” (32/64 bit) we also have two versions of the “NT Headers” data structure. We can see the hierarchy of the “NT Headers” in the output of “CFF Explorer” created by Erik Pistelli which allows viewing/modifying PE files¹⁵ - as shown in the screenshot below.



¹⁰ <https://medium.com/@boutnaru/the-portable-executable-journey-dos-header-ea5b29f15612>

¹¹ <https://medium.com/@boutnaru/the-portable-executable-journey-dos-stub-0ca8cda20570>

¹² https://learn.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-image_nt_headers32

¹³ https://learn.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-image_nt_headers64

¹⁴ <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format#signature-image-only>

¹⁵ <https://ntcore.com/explorer-suite/>

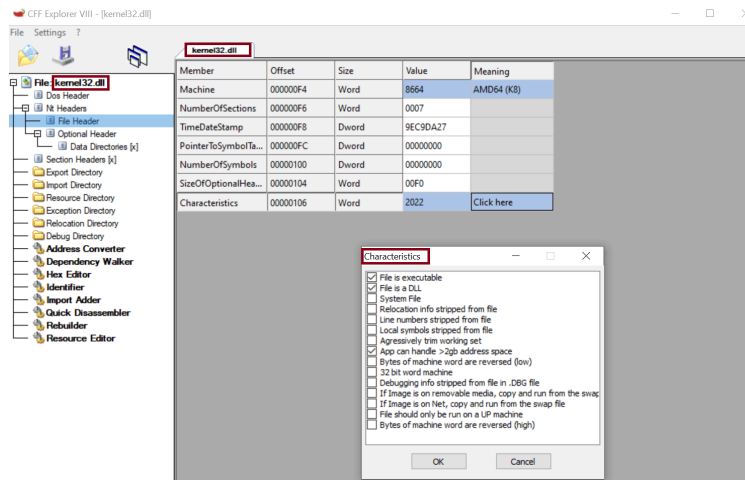
File Header (struct _IMAGE_FILE_HEADER)

“File Header” represents the COFF (Common Object File Format) header. COFF is used for storing compiled code (output of a compiler and linker). Thus, PE (sometimes called PE/COFF) contains a version of COFF. Its goal is to hold basic information about the file while containing pointers to other data structures. By the way, this header is fixed in size¹⁶.

Overall, the “File Header” consists of seven fields: “Machine”, “NumberOfSections”, “TimeDateStamp”, “PointerToSymbolTable”, “NumberOfSymbols”, “SizeOfOptionalHeader” and “Characteristics”. The “Machine” field describes the architecture which the file can be executed on, examples of relevant values are: x86 (IMAGE_FILE_MACHINE_I386 which equals to 0x014c), “Intel Itanium” (IMAGE_FILE_MACHINE_IA64 which equals to 0x0200) and “x64” (IMAGE_FILE_MACHINE_AMD64 which equals to 0x8664) - more values are available in the Microsoft documentation¹⁷. “NumberOfSections” defines how long is the section table (by the way the limit of the Windows loader is 96). “TimeDateStamp” is the number of seconds (since January 1, 1970 midnight) according to the system clock inserted by the linker¹⁸.

Moreover, “PointerToSymbolTable” is the offset in bytes to the symbol (name to address of variable/function) table (0 if there is no symbol table). “NumberOfSymbols” is the number of symbols in the symbol table. “SizeOfOptionalHeader” is the size of the “Optional Header” in bytes. By the way, if the size is “0” the file is an “object file”¹⁹.

Lastly, the “Characteristics” field is used to describe the attributes of the file like: support of >2GB addresses, debugging information is removed, the file is a DLL and more²⁰. An example of parsing the “File Header” for the “kernel32.dll” file using “CFF Explorer” is shown below.



¹⁶ <https://wiki.osdev.org/COFF>

¹⁷ <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format#machine-types>

¹⁸ https://learn.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-image_file_header

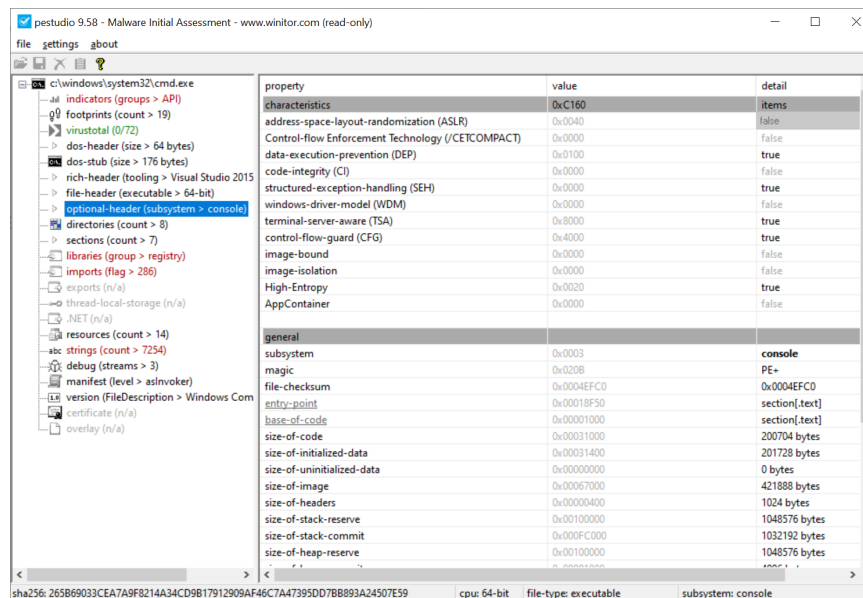
¹⁹ <https://gist.github.com/TheWover/730275aedcb4a2413cdbc8a2e33a8df4>

²⁰ <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format#characteristics>

Optional Header (struct `_IMAGE_OPTIONAL_HEADER32/64`)

Despite its name the “Optional Header” is not optional (it is required) in case of a compiled and linked binary. This header contains information used by the OS loader when loading a PE file. Because we can have a 64-bit or 32-bit PE file, there are corresponding versions of the “Optional Header” data structure: “struct `_IMAGE_OPTIONAL_HEADER64`” and “struct `_IMAGE_OPTIONAL_HEADER32`”²¹.

Moreover, remember that the size of the “Optional Header” is not fixed and it is defined in the field “`SizeOfOptionalHeader`” as part of the “File Header”²². They are multiple fields as part of the “Optional Header” - as shown in the screenshot below (taken using pestudio). In general we can divide the optional header to three main parts: standard fields, Windows specific fields and data directories²³.



Thus, the standard fields part includes eight fields. “Magic”, which can be “0x10b” (PE) and “0x20b” (PE+), the second allows access to 64-bit address space. “MajorLinkerVersion” and “MinorLinkerVersion” which are the major and minor versions of the linker. “SizeOfCode” which is the total size of all code sections (.text and more if relevant). “SizeOfInitializedData” and “SizeOfUninitializedData” which is the total size of initialized and uninitialized (BSS) sections. “AddressOfEntryPoint” and “BaseOfCode”²⁴.

²¹ <https://github.com/reactos/reactos/blob/master/sdk/include/host/pecoff.h#L139>

²² <https://medium.com/@boutnaru/the-portable-executable-journey-file-header-struct-image-file-header-00360271f147>

²³ <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format>

²⁴ <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format>

Lastly, the Windows specific field part contains 21 fields. “ImageBase”, “SectionAlignment”, “FileAlignment”, “MajorOperatingSystemVersion”, “MinorOperatingSystemVersion”, “MajorImageVersion”, “MinorImageVersion”, “MajorSubsystemVersion”, “MinorSubsystemVersion”, “Win32VersionValue” (reserved, must be “0”), “SizeOfImage”, “SizeOfHeaders”, “Checksum”, “Subsystem”, “DllCharacteristics”, “SizeOfStackReserve”, “SizeOfStackCommit”, “SizeOfHeapReserve”, “SizeOfHeapCommit”, “LoaderFlags” (reserved, must be zero) and “NumberOfRvaAndSizes”²⁵.

²⁵ https://learn.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-image_optional_header64